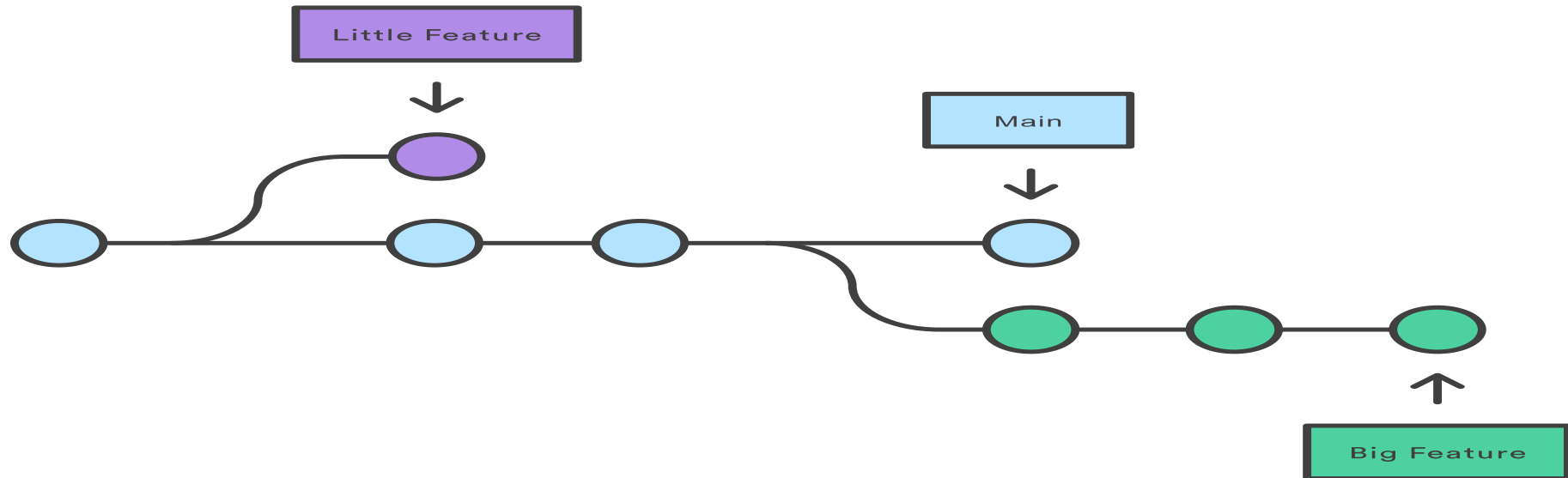


# ramas de git

La creación de ramas es una función disponible en la mayoría de los sistemas de control de versiones modernos. La creación de ramas en otros sistemas de control de versiones puede tanto llevar mucho tiempo como ocupar mucho espacio de almacenamiento. En Git, las ramas son parte del proceso de desarrollo diario. Las ramas de Git son un puntero eficaz para las instantáneas de tus cambios. Cuando quieres añadir una nueva función o solucionar un error, independientemente de su tamaño, generas una nueva rama para alojar estos cambios. Esto hace que resulte más complicado que el código inestable se fusione con el código base principal, y te da la oportunidad de limpiar tu historial futuro antes de fusionarlo con la rama principal.



# ramas de git

El diagrama anterior representa un repositorio con dos líneas de desarrollo aisladas, una para una función pequeña y otra para una función más extensa. Al desarrollarlas en ramas, no solo es posible trabajar con las dos de forma paralela, sino que también se evita que el código dudoso se fusione con la rama main

La implementación que subyace a las ramas de Git es mucho más sencilla que la de otros modelos de sistemas de control de versiones. En lugar de copiar archivos entre directorios, Git almacena una rama como referencia a una confirmación. En este sentido, una rama representa el extremo de una serie de confirmaciones, es decir, no es un contenedor de confirmaciones.

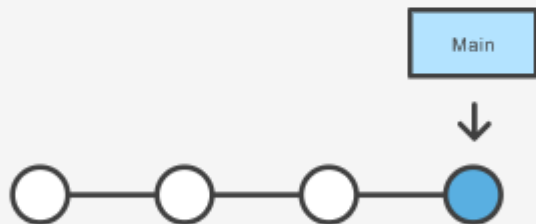
# Funcionamiento

Una rama representa una línea independiente de desarrollo. Las ramas sirven como una abstracción de los procesos de cambio, preparación y confirmación. Puedes concebirlas como una forma de solicitar un nuevo directorio de trabajo, un nuevo entorno de ensayo o un nuevo historial de proyecto

El comando `git branch` te permite crear, enumerar y eliminar ramas, así como cambiar su nombre. No te permite cambiar entre ramas o volver a unir un historial bifurcado. Por este motivo, `git branch` está estrechamente integrado con los comandos `git checkout` y `git merge`.

# Creación de ramas

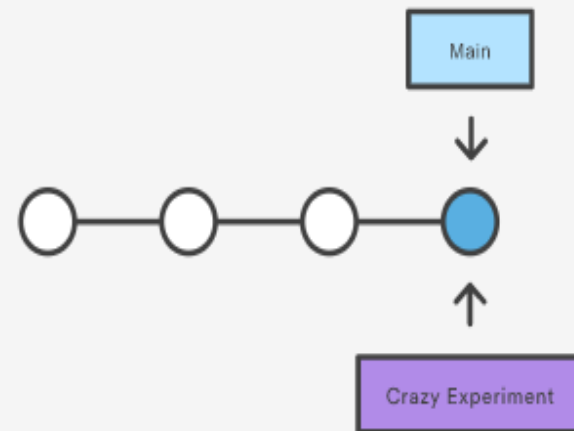
Es importante comprender que las ramas son solo punteros a las confirmaciones. Cuando creas una rama, todo lo que Git tiene que hacer es crear un nuevo puntero, no modifica el repositorio de ninguna otra forma. Si empiezas con un repositorio que tiene este aspecto:



Y, a continuación, creas una rama con el siguiente comando:

```
git branch crazy-experiment
```

El historial del repositorio no se modificará. Todo lo que necesitas es un nuevo puntero de la confirmación actual:



Ten en cuenta que este comando solo *crea* la nueva rama. Para empezar a añadir confirmaciones, necesitas seleccionarla con el comando `git checkout` y, a continuación, utilizar los comandos estándar `git add` y `git commit`.

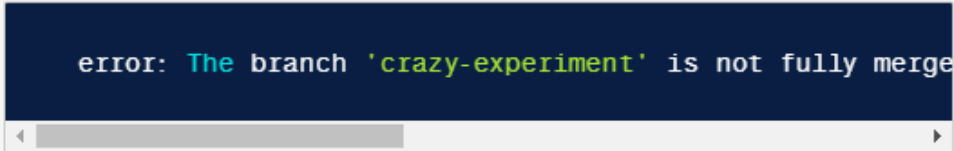
# Eliminación de ramas

Una vez que hayas terminado de trabajar en una rama y la hayas fusionado con el código base principal, puedes eliminar la rama sin perder ninguna historia:

```
git branch -d crazy-experiment
```

No obstante, si la rama no se ha fusionado, el comando anterior mostrará un mensaje de error:

```
error: The branch 'crazy-experiment' is not fully merge
```



Esto te protege ante la pérdida de acceso a una línea de desarrollo completa. Si realmente quieres eliminar la rama (por ejemplo, si se trata de un experimento fallido), puedes usar el indicador `-D` (en mayúscula):

```
git branch -D crazy-experiment
```

Este comando elimina la rama independientemente de su estado y sin avisos previos, así que úsalo con cuidado.

# Cambio de rama

El comando `git checkout` te permite desplazarte entre las ramas creadas por `git branch`. Al extraer una rama, se actualizan los archivos en el directorio de trabajo para reflejar la versión almacenada en esa rama y se indica a Git que registre todas las confirmaciones nuevas en dicha rama. Puedes contemplar todo esto como una forma de seleccionar la línea de desarrollo en la que trabajas.

Disponer de una rama específica para cada nueva función supone un cambio drástico en comparación con el flujo de trabajo tradicional de SVN. Hace que resulte ridículamente sencillo probar nuevos experimentos sin temor a arruinar las funciones existentes y permite trabajar al mismo tiempo en muchas funciones que no guardan relación entre sí. Además, las ramas facilitan varios flujos de trabajo colaborativos.

En ocasiones, el comando `git checkout` puede confundirse con `git clone`. La diferencia entre ambos comandos estriba en que el segundo recupera código de un repositorio remoto, mientras que el primero cambia entre versiones de código que ya se encuentran en el sistema local.

# Git merge

La fusión es la forma que tiene Git de volver a unir un historial bifurcado. El comando `git merge` permite tomar las líneas independientes de desarrollo creadas por `git branch` e integrarlas en una sola rama.

Ten en cuenta que todos los comandos presentados a continuación se fusionan en la rama actual. La rama actual se actualizará para reflejar la fusión, pero la rama de destino no se verá afectada en absoluto. Una vez más, esto significa que `git merge` se suele utilizar junto con `git checkout` para seleccionar la rama actual y `git branch -d` para eliminar la rama de destino obsoleta.